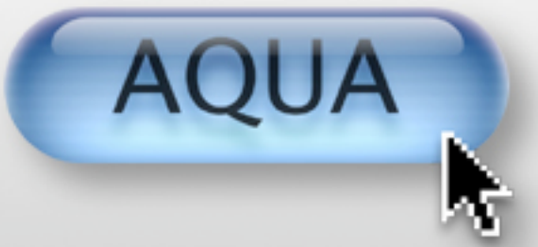
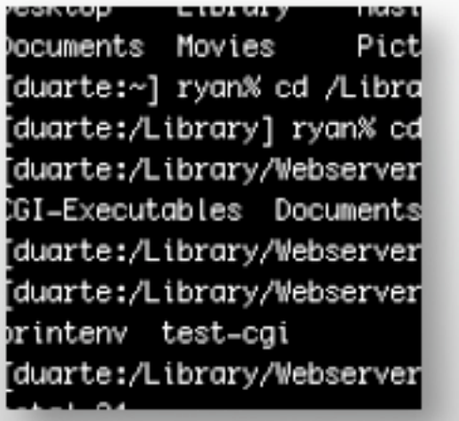
HIView In-Depth

Session 425





HIView In-Depth

Ed Voas
Manager/Tech Lead, HIToolbox

What You'll Learn

- Quick overview of HIView
- How to use HIView
- How to write your own HIViews
- How to make your views perform well

What Is HIView?

- Complete replacement for Control Manager
  - Pervasive—Menus, Windows, Controls
- Modern
  - Composited, one-pass draw model
  - Floating-point coordinate system
  - Quartz drawing
  - Overlapping views
  - Layout facilities (new in Panther)

Composited Drawing

- Predictable Drawing Order
  - Back to front
  - Respects z-order of siblings
- No erase-behind
  - Pattern alignment issues disappear!
- Drawing occurs at event loop time
  - You invalidate the areas you want redrawn

Invalidation

- Invalidation occurs automatically on move/resize only
  - On resize, you can control what gets invalidated—kControlInvalidateForSizeChange
- You must invalidate manually otherwise
- Invalidating parent does not invalidate children!

```
OSStatus HIViewSetNeedsDisplay(
   HIViewRef          inView,
   Boolean            inNeedsDisplay );

OSStatus HIViewSetNeedsDisplayInRegion(
   HIViewRef          inView,
   RgnHandle          inRegion,
   Boolean            inNeedsDisplay );
```

Geometry

Geometry Types

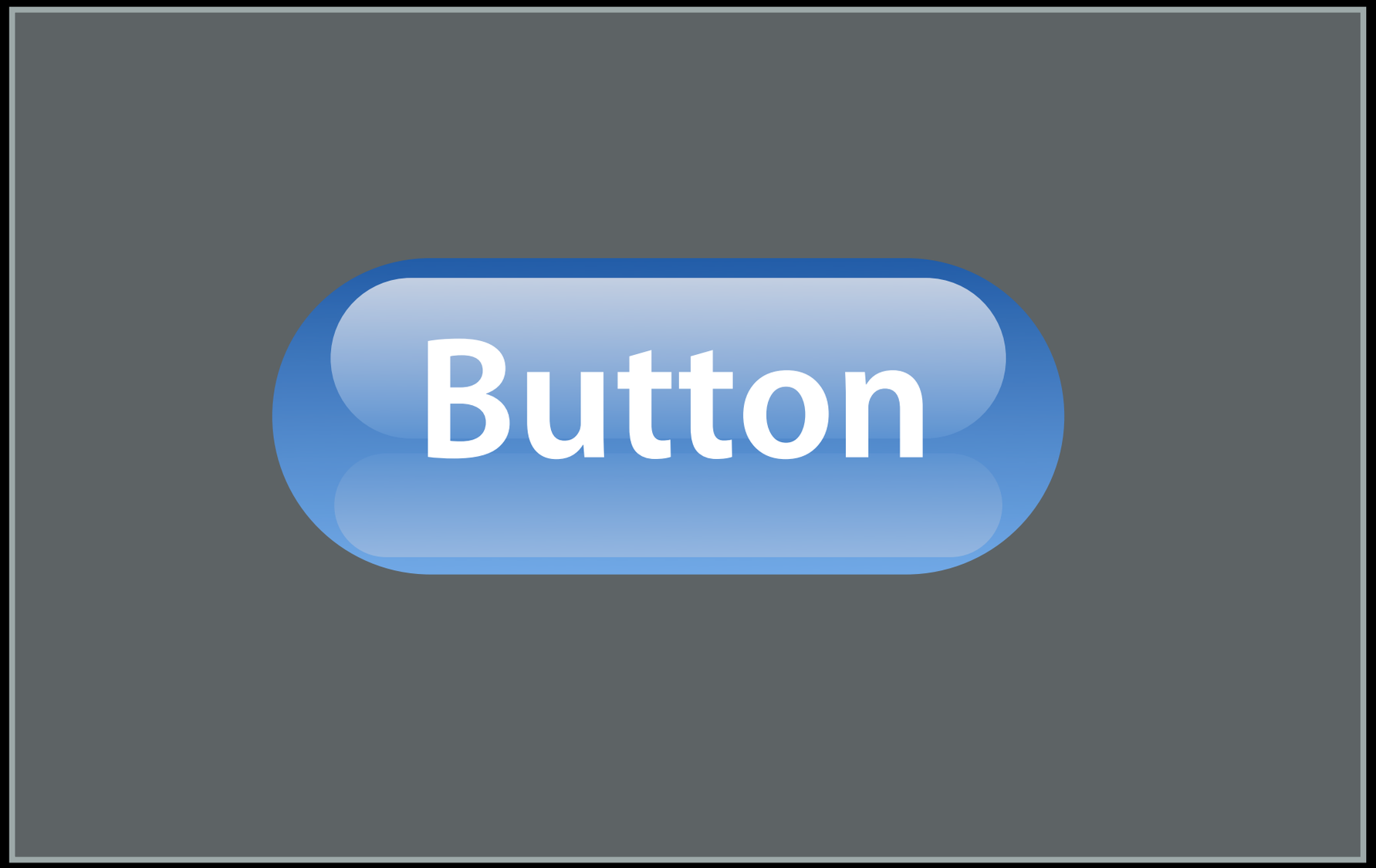
- Floating-point types to replace QD types
  - HIPoint
  - HISize
  - HIRect
- All HIView APIs are in terms of these types

Coordinate Systems

- Two rectangles for every view
  - Frame: Position within parent view
  - Bounds: View's local coordinates
- You move/resize your view based on Frame
- You draw and hit test based on Bounds

HIView Bounds

Group Box



HIView Bounds

Group Box

HIView Frame

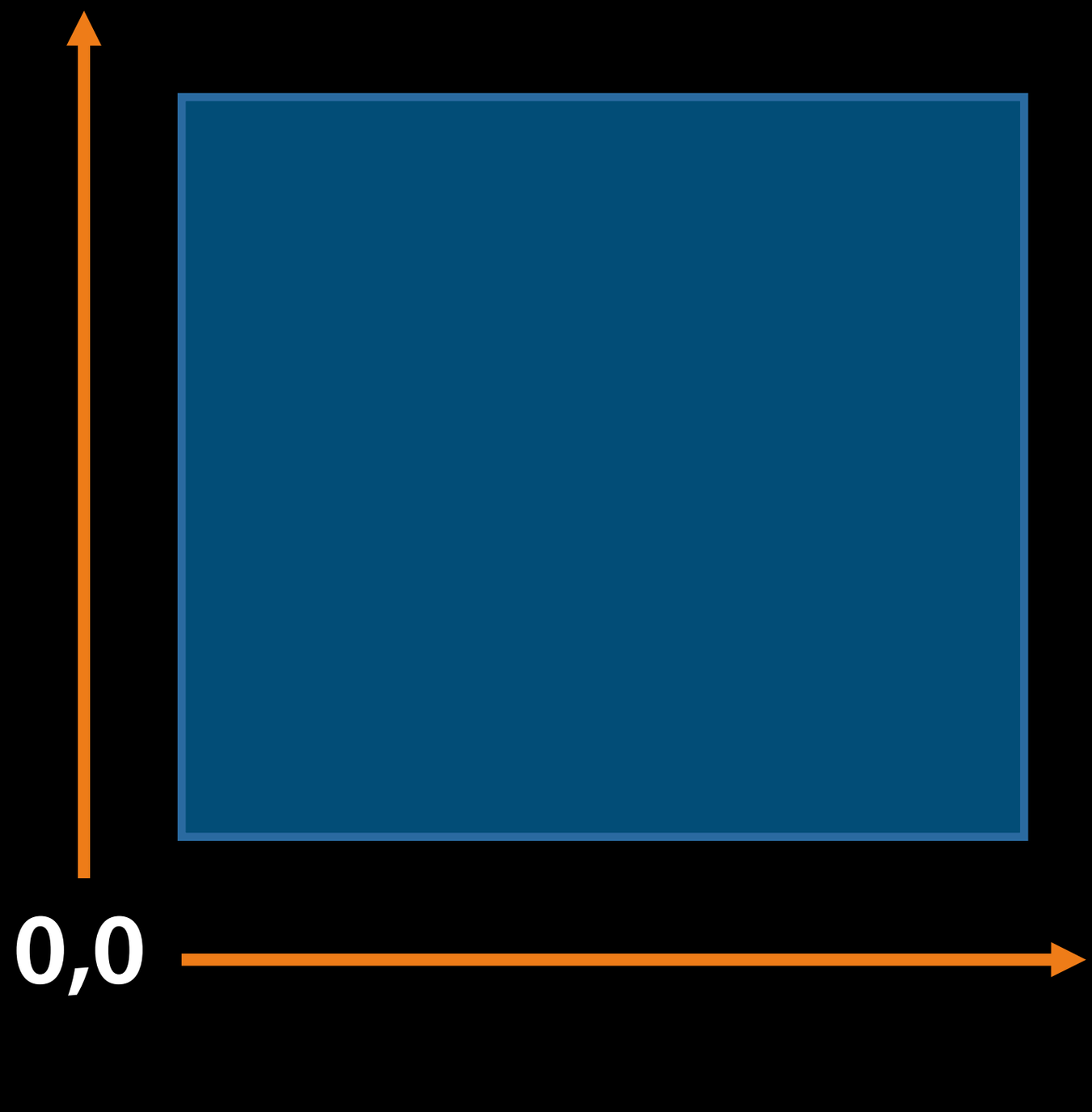
Group Box

Graphics Models

Two Graphics Models

- Quartz
  - Native graphics model
  - Transparency, anti-aliasing
  - Beziers, scaling, rotating, etc.
- QuickDraw
  - For quick migration

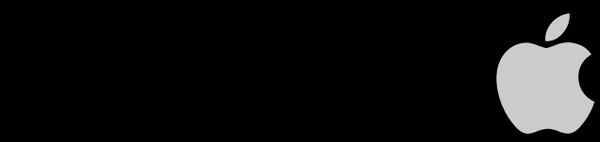
Quartz Coordinates



HIView Coordinates

Using Quartz

- Context is transformed—0, 0 at top left
  - Windowing system is top left
  - Your existing code is top left
  - Don’t need to deal with window resize issues
  - BUT, you are drawing upside down
- Requires some new APIs to draw some things
  - HIViewDrawCGImage
  - HIThemeDrawText



Layout

Layout Facilities

- Automatic positioning and sizing of views based on other views
- Multiple aspects
  - Bindings, Positions, Scalings
- Can relate position/size of view to any other view
  - HIViewGet/SetLayoutInfo
  - HIViewApplyLayout

Layout Sample

- Set side bindings

```
layoutInfo.version = kHILayoutInfoVersionZero;
err = HIViewGetLayoutInfo( view, &layoutInfo );

layoutInfo.bindings.left.kind = kHILayoutBindLeft;
layoutInfo.bindings.left.toView = NULL; // to parent
layoutInfo.bindings.right.kind = kHILayoutBindRight;
layoutInfo.bindings.right.toView = NULL; // to parent

err = HIViewSetLayoutInfo( view, &layoutInfo );
```

Layout Sample

- Set scaling

```
layoutInfo.version = kHILayoutInfoVersionZero;
err = HIViewGetLayoutInfo( view, &layoutInfo );

layoutInfo.scale.x.kind = kScaleAbsolute;
layoutInfo.scale.x.ratio = 0.5;
layoutInfo.scale.x.toView = NULL; // to parent

err = HIViewSetLayoutInfo( view, &layoutInfo );
```

Layout Sample

- Set position

```
layoutInfo.version = kHILayoutInfoVersionZero;
err = HIViewGetLayoutInfo( view, &layoutInfo );

layoutInfo.position.x.kind = kHILayoutPositionCenter;
layoutInfo.position.x.toView = theViewAboveMe;
layoutInfo.position.x.offset = 0;

err = HIViewSetLayoutInfo( view, &layoutInfo );
```

Demo

Using HIView

Turn On Compositing

- Per-window
- Must use CreateNewWindow
- kWindowCompositingAttribute
  - Specifiable only at window creation time
- Use the standard window handler

```
err = CreateNewWindow( kDocumentWindowClass,
   kWindowStandardDocumentAttributes |
   kWindowStandardHandlerAttribute |
   kWindowCompositingAttribute,
   &bounds, &myNewWindow );
```

Hierarchy Changes

- With compositing on, hierarchy is deeper
- Root view is the window frame
  - Window widgets are views themselves
  - Contains a content view
- From root you can find any standard subview
  - HIViewFindByID

Effect On CreateRootControl

- In compositing mode, CreateRootControl will fail
  - errRootAlreadyExists
- GetRootControl will yield the content root
  - Maintains compatible behavior
- HIViewGetRoot will yield the true root

```
GetRootControl( window, &contentView );

HIViewFindByID(      HIViewGetRoot( window ),
                     kHIViewWindowContentID,
                     &contentView );
```

A Simple Scenario

- You have a window with all Toolbox controls
  - Just pass the kWindowCompositingAttribute
- You are done (mostly)

Common Pitfalls

- Get/SetControlBounds work in HIView Frame space
  - Examine your uses, replace with appropriate call
  - Use HIViewConvertPoint/Rect to convert types
- Embedding using port coordinates
  - Make sure to convert into HIView Frame space

Do’s and Don’ts

- If you previously called Draw1Control/UpdateControls
  - Call HIViewSetNeedsDisplay
  - If you must draw immediately: HIViewRender (Panther)
- If you previously called DrawControlInCurrentPort
  - Don’t. Bad things will happen in compositing mode.
  - HIViewCreateOffscreenImage might help
- If you previously called AutoEmbedControl
  - Use HIViewAddSubview (or even EmbedControl)

Dialog Manager

Q: I have a Dialog Manager-based dialog
How do I make that support HIView?

A: Switch over to using nibs
(which gives you other benefits)

Writing Your Own HIViews

All Your Views Are Belong to Us

- Custom content must be converted to an HIView
- Required to ensure consistent behavior

Basic Steps to Write an HIView

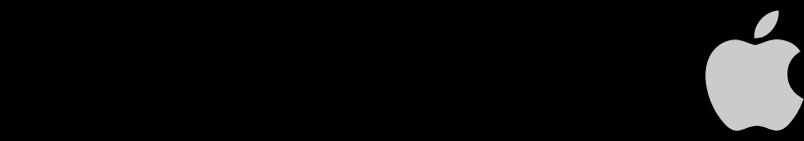
- Register your view class via HIObject mechanism
  - You can subclass HIView itself or another view class
- Instantiate your view via HIObjectCreate
- Handle events as they are sent to you

The Draw Event

- Carbon Event: kEventControlDraw
- Two parameters
  - CGContextRef: where to put the pixels
  - Draw region: how much to draw

Calculating Regions

- Carbon Event: kEventControlGetRegion
- When the Toolbox needs regions, it asks your view
- Regions should be in HIView Bounds coordinates
- Two most important regions
  - Structure region
  - Opaque region
- Other Interesting Regions
  - Clickable region (new in Panther)



Structure Region

- This can extend outside your Bounds
  - Focus rings
- If you don’t respond, your structure is your Bounds
- If you ever need to reshape yourself
  - HIViewReshapeStructure
  - Recomputes your structure and invalidates

```
OSStatus HIViewReshapeStructure(
   HIViewRef          inView );
```

Opaque Region

- Helps determine what is visible below you
  - Optimizes drawing
- Helps determine window's opaque region
- If not handled, you are considered transparent
- If you change your opaque region:

```
OSStatus HIViewReshapeStructure(
   HIViewRef          inView );

OSStatus HIViewRegionChanged(
   HIViewRef          inView,
   ControlPartCode    inRegionCode );
```

Region Smarts

- It’s hip to be square!
  - Better to have a square region than a complex one
- Use alpha to control what gets drawn
- Use complex regions only for hit testing

Features

- Advertise abilities of a view
  - kHIViewAllowsSubviews
  - kHIViewIsOpaque
- On Jaguar
  - Handle kEventControlInitialize, and return features
- On Panther
  - Use HIViewChangeFeatures
  - Works on-the-fly

Three Interesting Features

- kHIViewIsOpaque
  - Marks entire view as opaque
  - No need to deal with returning an opaque region
- kHIViewIgnoresClicks
  - Clicks pass through to views behind you
- kHIViewDoesNotDraw
  - View does not draw
  - Saves work

Code and Demo

Performance Tips

- Invalidate only what needs to be redrawn
- Obey the incoming limit region when drawing
- Watch content view bounds changed
  - Not Window bounds changed
  - Best approach—use layouts!
- Mark yourself with features if possible
  - kHIViewDoesNotDraw
  - kHIViewIsOpaque

Wrap Up

Takeaways

- HIView is our focus
  - Everything we do is HIView-based
- It frees you from the drudgery of the past
  - But yet familiar and easy to use
- Simplifies by unifying our xDEF models
- Allows us to make improvements that will directly benefit your applications
- It’s Fast. It’s Furious.

Roadmap

409 Carbon: The HIToolbox Learn about new features of Panther HIToolbox	**The Presidio** **Wed., 9:00am**
416 Carbon: Optimizing Performance Make your Carbon app scream!	**The Presidio** **Thu., 2:00pm**
415 Data Exchange For Carbon Apps Learn about new pasteboard APIs	**Russian Hill** **Thu., 10:30am**
423 Making a Modern Carbon Application See the best practices for developing Carbon apps	**Nob Hill** **Fri., 2:00pm**

Who to Contact

Xavier Legros
Evangeliste de Mac OS X
xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Introducing HIView

http://developer.apple.com/documentation/Carbon/Conceptual/HIViewDoc/index.html

HIView Reference

http://developer.apple.com/documentation/Carbon/Reference/HIViewReference/index.html

HIObject Reference

http://developer.apple.com/documentation/Carbon/Reference/HIObjectReference/index.html

HIToolbar Reference

http://developer.apple.com/documentation/Carbon/Reference/HIToolbarReference/index.html

Reference Library

Documentation (continued)

Using HIToolbar

http://developer.apple.com/documentation/Carbon/Conceptual/UsingHIToolbar/index.html

Reference Library

Sample Code

HIViewTest (and others)

ADC Home > Sample Code > Human Interface Toolbox > Mac OS X HIToolbox

MenuViews (new!)

ADC Home > Sample Code

Q&A

Xavier Legros

Mac OS X Evangelist, Crazy French Guy

xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference